

ALGOL (/ˈælɡoʊl, -ɡɔːl/; short for "**Algorithmic Language**")^[1] is a family of imperative computer programming languages originally developed in 1958. ALGOL heavily influenced many other languages and was the standard method for algorithm description used by the Association for Computing Machinery (ACM) in textbooks and academic sources until object-oriented languages came around, for more than thirty years.^[2]

In the sense that the syntax of most modern languages is "Algol-like",^[3] it was arguably the most influential of the four high-level programming languages among which it was roughly contemporary: FORTRAN, Lisp, and COBOL.^[4] It was designed to avoid some of the perceived problems with FORTRAN and eventually gave rise to many other programming languages, including PL/I, Simula, BCPL, B, Pascal, and C.

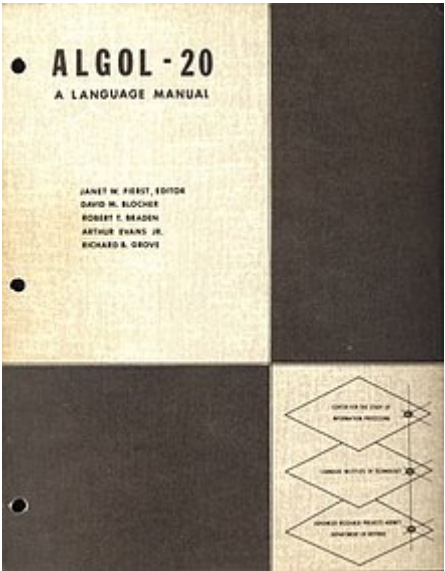
ALGOL introduced code blocks and the begin...end pairs for delimiting them. It was also the first language implementing nested function definitions with lexical scope. Moreover, it was the first programming language which gave detailed attention to formal language definition and through the *Algol 60 Report* introduced Backus–Naur form, a principal formal grammar notation for language design.

There were three major specifications, named after the years they were first published:

- ALGOL 58 – originally proposed to be called *IAL*, for *International Algebraic Language*.
- ALGOL 60 – first implemented as *X1 ALGOL 60* in mid-1960. Revised 1963.^{[5][6]}
- ALGOL 68 – introduced new elements including flexible arrays, slices, parallelism, operator identification. Revised 1973.^[7]

ALGOL 68 is substantially different from ALGOL 60 and was not well received, so that in general "Algol" means ALGOL 60 and dialects thereof.

Contents
Important implementations
History
Algol and programming language research

ALGOL	
	
A 1965 manual for ALGOL-20	
Paradigm	Procedural, imperative, structured
Family	ALGOL
Designed by	Bauer, Bottenbruch, Rutishauser, Samelson, Backus, Katz, Perlis, Wegstein, Naur, Vauquois, van Wijngaarden, Woodger, Green, McCarthy
First appeared	1958
Typing discipline	Static, strong
Scope	Lexical
Influenced	
Most subsequent imperative languages (so-called <i>ALGOL-like</i> languages) e.g. PL/I, Simula, BCPL, B, Pascal, C	

IAL implementations timeline

Properties

Examples and portability issues

Code sample comparisons

ALGOL 60

ALGOL 68

Timeline: Hello world

ALGOL 58 (IAL)

ALGOL 60 family

ALGOL 68

Timeline of ALGOL special characters

See also

References

Further reading

External links

Important implementations

The International Algebraic Language (IAL), renamed ALGOL 58, was highly influential and generally considered the ancestor of most of the modern programming languages (the so-called Algol-like languages). Further, *ALGOL object code* was a simple, compact, and stack-based instruction set architecture commonly used in teaching compiler construction and other high order languages; of which Algol is generally considered the first.

History

ALGOL was developed jointly by a committee of European and American computer scientists in a meeting in 1958 at the Swiss Federal Institute of Technology in Zurich (ETH Zurich; cf. ALGOL 58). It specified three different syntaxes: a reference syntax, a publication syntax, and an implementation syntax. The different syntaxes permitted it to use different keyword names and conventions for decimal points (commas vs periods) for different languages.

ALGOL was used mostly by research computer scientists in the United States and in Europe. Its use in commercial applications was hindered by the absence of standard input/output facilities in its description and the lack of interest in the language by large computer vendors other than Burroughs Corporation. ALGOL 60 did however become the standard for the publication of algorithms and had a profound effect on future language development.

John Backus developed the *Backus normal form* method of describing programming languages specifically for ALGOL 58. It was revised and expanded by Peter Naur for ALGOL 60, and at Donald Knuth's suggestion renamed Backus–Naur form.^[8]

Peter Naur: "As editor of the ALGOL Bulletin I was drawn into the international discussions of the language and was selected to be member of the European language design group in November 1959. In this capacity I was the editor of the ALGOL 60 report, produced as the result of the ALGOL 60 meeting in Paris in January 1960."^[9]

The following people attended the meeting in Paris (from 1 to 16 January):

- Friedrich L. Bauer, Peter Naur, Heinz Rutishauser, Klaus Samelson, Bernard Vauquois, Adriaan van Wijngaarden, and Michael Woodger (from Europe)
- John W. Backus, Julien Green, Charles Katz, John McCarthy, Alan J. Perlis, and Joseph Henry Wegstein (from the USA).

Alan Perlis gave a vivid description of the meeting: "The meetings were exhausting, interminable, and exhilarating. One became aggravated when one's good ideas were discarded along with the bad ones of others. Nevertheless, diligence persisted during the entire period. The chemistry of the 13 was excellent."

ALGOL 60 inspired many languages that followed it. Tony Hoare remarked: "Here is a language so far ahead of its time that it was not only an improvement on its predecessors but also on nearly all its successors."^[10] The Scheme programming language, a variant of Lisp that adopted the block structure and lexical scope of ALGOL, also adopted the wording "Revised Report on the Algorithmic Language Scheme" for its standards documents in homage to ALGOL.^[11]

Algol and programming language research

As Peter Landin noted, the language Algol was the first language to combine seamlessly imperative effects with the (call-by-name) lambda calculus. Perhaps the most elegant formulation of the language is due to John C. Reynolds, and it best exhibits its syntactic and semantic purity. Reynolds's idealized Algol also made a convincing methodological argument regarding the suitability of local effects in the context of call-by-name languages, to be contrasted with the global effects used by call-by-value languages such as ML. The conceptual integrity of the language made it one of the main objects of semantic research, along with Programming Computable Functions (PCF) and ML.^[12]

IAL implementations timeline

To date there have been at least 70 augmentations, extensions, derivations and sublanguages of Algol 60.^[13]



Family tree of the Algol, Fortran and COBOL programming language dynasty

Name	Year	Author	Country	Description	Target CPU
ZMMD-implementation	1958	Friedrich L. Bauer, Heinz Rutishauser, Klaus Samelson, Hermann Bottenbruch	Germany	implementation of <u>ALGOL 58</u>	<u>Z22</u> (later Zuse's <u>Z23</u> was delivered with an Algol 60 compiler) ^[14]
X1 ALGOL 60	August 1960 ^[15]	<u>Edsger W. Dijkstra</u> and <u>Jaap A. Zonneveld</u>	Netherlands	First implementation of ALGOL 60 ^[16]	<u>Electrologica X1</u>
<u>Elliott ALGOL</u>	1960s	<u>C. A. R. Hoare</u>	UK	Subject of the 1980 <u>Turing</u> lecture ^[17]	Elliott 803, Elliott 503, Elliott 4100 series
<u>JOVIAL</u>	1960	<u>Jules Schwartz</u>	USA	A <u>DOD HOL</u> prior to <u>Ada</u>	Various (see article)
Burroughs Algol (Several variants)	1961	Burroughs Corporation (with participation by Hoare, <u>Dijkstra</u> , and others)	USA	Basis of the Burroughs (and now <u>Unisys MCP</u> based) computers	Burroughs large systems and their midrange also.
<u>Case ALGOL</u>	1961	<u>Case Institute of Technology</u> ^[18]	USA	<u>Simula</u> was originally contracted as a simulation extension of the Case ALGOL	<u>UNIVAC 1107</u>
<u>GOGOL</u>	1961	<u>William M. McKeeman</u>	USA	For ODIN time-sharing system ^[19]	<u>PDP-1</u>
<u>RegneCentralen ALGOL</u>	1961	Peter Naur, <u>Jørn Jensen</u>	Denmark	Implementation of full Algol 60	DASK at Regnecentralen
<u>Dartmouth ALGOL 30</u>	1962	<u>Thomas Eugene Kurtz</u> et al.	USA		<u>LGP-30</u>
<u>USS 90 Algol</u>	1962	<u>L. Petrone</u>	Italy		
Algol Translator	1962	G. van der Mey and <u>W.L. van der Poel</u>	Netherlands	Staatsbedrijf der Posterijen, Telegrafie en Telefonie	<u>ZEBRA</u>
<u>Kidsgrove Algol</u>	1963	<u>F. G. Duncan</u>	UK		<u>English Electric Company KDF9</u>
<u>VALGOL</u>	1963	<u>Val Schorre</u>	USA	A test of the <u>META II</u> compiler compiler	
<u>Whetstone</u>	1964	<u>Brian Randell</u> and L. J. Russell	UK	Atomic Power Division of English Electric Company. Precursor to <u>Ferranti Pegasus</u> , National Physical Laboratories ACE and <u>English Electric DEUCE</u> implementations.	<u>English Electric Company KDF9</u>
<u>NU ALGOL</u>	1965		Norway		<u>UNIVAC</u>
ALGEK	1965		<u>USSR</u>	<u>АЛГЭК</u> , based on	<u>Minsk-22</u>

				ALGOL-60 and COBOL support, for economical tasks	
<u>ALGOL W</u>	1966	<u>Niklaus Wirth</u>	USA	Proposed successor to ALGOL 60	<u>IBM System/360</u>
<u>MALGOL</u>	1966	publ. A. Viil, M Kotli & M. Rakhendi,	<u>Estonian SSR</u>		<u>Minsk-22</u>
<u>ALGAMS</u>	1967	GAMS group (ГАМС, группа автоматизации программирования для машин среднего класса), cooperation of Comecon Academies of Science	<u>Comecon</u>		<u>Minsk-22, later ES EVM, BESM</u>
<u>ALGOL/ZAM</u>	1967		Poland		Polish <u>ZAM</u> computer
<u>Simula 67</u>	1967	<u>Ole-Johan Dahl</u> and <u>Kristen Nygaard</u>	Norway	Algol 60 with classes	<u>UNIVAC 1107</u>
<u>Chinese Algol</u> (https://web.archive.org/web/20080722231533/http://hopl.murdoch.edu.au/showlanguage.prx?exp=7288&language=Chinese%20Algol)	1972		China	Chinese characters, expressed via the Symbol system	
<u>DG/L</u>	1972		USA		<u>DG Eclipse</u> family of Computers
<u>S-algol</u>	1979	<u>Ron Morrison</u>	UK	Addition of orthogonal datatypes with intended use as a teaching language	<u>PDP-11</u> with a subsequent implementation on the <u>Java VM</u>

The Burroughs dialects included special Bootstrapping dialects such as ESPOL and NEWP. The latter is still used for Unisys MCP system software.

Properties

ALGOL 60 as officially defined had no I/O facilities; implementations defined their own in ways that were rarely compatible with each other. In contrast, ALGOL 68 offered an extensive library of *transput* (input/output) facilities.

ALGOL 60 allowed for two evaluation strategies for parameter passing: the common call-by-value, and call-by-name. Call-by-name has certain effects in contrast to call-by-reference. For example, without specifying the parameters as *value* or *reference*, it is impossible to develop a procedure that will swap the values of two parameters if the actual parameters that are passed in are an integer variable and an array that is indexed by that same integer variable.^[20] Think of passing a pointer to swap(*i*, A[i]) in to a function. Now that every time swap is referenced, it is reevaluated. Say *i* := 1 and A[i] := 2, so every time swap is referenced it will return the other combination of the values ([1,2], [2,1], [1,2] and so on). A similar situation occurs with a random function passed as actual argument.

Call-by-name is known by many compiler designers for the interesting "thunks" that are used to implement it. Donald Knuth devised the "man or boy test" to separate compilers that correctly implemented "recursion and non-local references." This test contains an example of call-by-name.

ALGOL 68 was defined using a two-level grammar formalism invented by Adriaan van Wijngaarden and which bears his name. Van Wijngaarden grammars use a context-free grammar to generate an infinite set of productions that will recognize a particular ALGOL 68 program; notably, they are able to express the kind of requirements that in many other programming language standards are labelled "semantics" and have to be expressed in ambiguity-prone natural language prose, and then implemented in compilers as *ad hoc* code attached to the formal language parser.

Examples and portability issues

Code sample comparisons

ALGOL 60

(The way the bold text has to be written depends on the implementation, e.g. 'INTEGER'—quotation marks included—for **integer**. This is known as stropping.)

```
procedure Absmax(a) Size:(n, m) Result:(y) Subscripts:(i, k);
  value n, m; array a; integer n, m, i, k; real y;
comment The absolute greatest element of the matrix a, of size n by m
  is transferred to y, and the subscripts of this element to i and k;
begin
  integer p, q;
  y := 0; i := k := 1;
  for p := 1 step 1 until n do
    for q := 1 step 1 until m do
      if abs(a[p, q]) > y then
        begin y := abs(a[p, q]);
              i := p; k := q
        end
  end Absmax
```

Here is an example of how to produce a table using Elliott 803 ALGOL.^[21]

```
FLOATING POINT ALGOL TEST'
BEGIN REAL A,B,C,D'

READ D'

FOR A:= 0.0 STEP D UNTIL 6.3 DO
BEGIN
  PRINT PUNCH(3),££L??'
  B := SIN(A)'
  C := COS(A)'
  PRINT PUNCH(3), SAMELINE, ALIGNED(1, 6), A, B, C'
END'
END'
```

PUNCH(3) sends output to the teleprinter rather than the tape punch.

SAMELINE suppresses the carriage return + line feed normally printed between arguments.

ALIGNED(1,6) controls the format of the output with 1 digit before and 6 after the decimal point.

ALGOL 68

The following code samples are ALGOL 68 versions of the above ALGOL 60 code samples.

ALGOL 68 implementations used ALGOL 60's approaches to stropping. In ALGOL 68's case tokens with the **bold** typeface are reserved words, types (**modes**) or operators.

```
proc abs max = ([,]real a, ref real y, ref int i, k)real:
comment The absolute greatest element of the matrix a, of size [a by 2[a
is transferred to y, and the subscripts of this element to i and k; comment
begin
  real y := 0; i := [a; k := 2[a;
  for p from [a to [a do
    for q from 2[a to 2[a do
      if abs a[p, q] > y then
        y := abs a[p, q];
        i := p; k := q
      fi
    od
  od;
  y
end # abs max #
```

Note: lower (**[**) and upper (**]**) bounds of an array, and array slicing, are directly available to the programmer.

```
floating point algol68 test:
(
  real a,b,c,d;

  # printf - sends output to the file stand out. #
  # printf($p$); - selects a new page #
  printf(($pg$, "Enter d:"));
  read(d);

  for step from 0 while a:=step*d; a <= 2*pi do
    printf($l$); # $l$ - selects a new line. #
    b := sin(a);
    c := cos(a);
    printf(($z-d.6d$, a,b,c)) # formats output with 1 digit before and 6 after the decimal
  point. #
  od
)
```

Timeline: Hello world

The variations and lack of portability of the programs from one implementation to another is easily demonstrated by the classic hello world program.

ALGOL 58 (IAL)

ALGOL 58 had no I/O facilities.

ALGOL 60 family

Since ALGOL 60 had no I/O facilities, there is no portable hello world program in ALGOL. The next three examples are in Burroughs Extended Algol. The first two direct output at the interactive terminal they are run on. The first uses a character array, similar to C. The language allows the array identifier to be used as a pointer to the array, and hence in a REPLACE statement.

```
BEGIN
  FILE F(KIND=REMOTE);
  EBCDIC ARRAY E[0:11];
  REPLACE E BY "HELLO WORLD!";
  WRITE(F, *, E);
END.
```

A simpler program using an inline format:

```
BEGIN
  FILE F(KIND=REMOTE);
  WRITE(F, <"HELLO WORLD!">);
END.
```

An even simpler program using the Display statement. Note that its output would end up at the system console ('SPO'):

```
BEGIN DISPLAY("HELLO WORLD!") END.
```

An alternative example, using Elliott Algol I/O is as follows. Elliott Algol used different characters for "open-string-quote" and "close-string-quote":

```
program HiFolks;
begin
  print 'Hello world';
end;
```

Here is a version for the Elliott 803 Algol (A104) The standard Elliott 803 used 5 hole paper tape and thus only had upper case. The code lacked any quote characters so £ (UK Pound Sign) was used for open quote and ? (Question Mark) for close quote. Special sequences were placed in double quotes (e.g. ££L?? produced a new line on the teleprinter).

```
HIFOLKS'
BEGIN
  PRINT £HELLO WORLD£L??'
END'
```

The ICT 1900 series Algol I/O version allowed input from paper tape or punched card. Paper tape 'full' mode allowed lower case. Output was to a line printer. The open and close quote characters were represented using '(' and ')' and spaces by %.^[22]

```
'BEGIN'
  WRITE TEXT('(' 'HELLO%WORLD')');
'END'
```

ALGOL 68

ALGOL 68 code was published with reserved words typically in lowercase, but bolded or underlined.

```
begin
  printf(($gl$, "Hello, world!"))
end
```

In the language of the "Algol 68 Report" the input/output facilities were collectively called the "Transput".

Timeline of ALGOL special characters

The ALGOLs were conceived at a time when character sets were diverse and evolving rapidly; also, the ALGOLs were defined so that only *uppercase* letters were required.

1960: IFIP – The Algol 60 language and report included several mathematical symbols which are available on modern computers and operating systems, but, unfortunately, were unsupported on most computing systems at the time. For instance: \times , \div , \leq , \geq , \neq , \neg , \vee , \wedge , \subset , \equiv , \sqsubset and ‰ .

1961 September: ASCII – The ASCII character set, then in an early stage of development, had the `\` (Back slash) character added to it in order to support ALGOL's boolean operators `^` and `v`.^[23]

1962: ALCOR – This character set included the unusual "*" runic cross^[24] character for multiplication and the "‰" Decimal Exponent Symbol^[25] for floating point notation.^{[26][27][28]}

1964: GOST – The 1964 Soviet standard GOST 10859 allowed the encoding of 4-bit, 5-bit, 6-bit and 7-bit characters in ALGOL.^[29]

1968: The "Algol 68 Report" – used extant ALGOL characters, and further adopted `→`, `↓`, `↑`, `□`, `⊥`, `⌈`, `⌋`, `⌊`, `⌋`, `⊙`, `⊥`, and `¢` characters which can be found on the IBM 2741 keyboard with *typeball* (or *golf ball*) print heads inserted (such as the APL golf ball). These became available in the mid-1960s while ALGOL 68 was being drafted. The report was translated into Russian, German, French, and Bulgarian, and allowed programming in languages with larger character sets, e.g., Cyrillic alphabet of the Soviet BESM-4. All ALGOL's characters are also part of the Unicode standard and most of them are available in several popular fonts.

2009 October: Unicode – The `‰` (Decimal Exponent Symbol) for floating point notation was added to Unicode 5.2 for backward compatibility with historic Buran programme ALGOL software.^[30]

See also

- Address programming language
- Atlas Autocode
- Coral 66
- Edinburgh IMP
- Jensen's Device
- ISWIM
- JOVIAL
- Tron (video game)
- NELIAC
- Simula
- S-algol
- Scheme (programming language)

References

1. The name of this language family is sometimes given in mixed case (*Algol 60* (<http://www.masswerk.at/algol60/report.htm>) Archived (<https://web.archive.org/web/20070625171638/http://www.masswerk.at/algol60/report.htm>) 25 June 2007 at the Wayback Machine), and sometimes in all uppercase (*ALGOL68* (<https://www.cs.ru.nl/~hubbers/courses/sl1/rr.pdf>) Archived (<https://web.archive.org/web/20140913132128/http://www.cs.ru.nl/~hubbers/courses/sl1/rr.pdf>) 13 September 2014 at the Wayback Machine). For simplicity this article uses *ALGOL*.
2. *Collected Algorithms of the ACM* (<http://calgo.acm.org/>) Archived (<http://archive.wikiwix.com/cache/20111017235805/http://calgo.acm.org/>) 17 October 2011 at Wikiwix Compressed archives of the algorithms. ACM.

3. O'Hearn, P. W.; Tennent, R. D. (September 1996). "Algol-like languages, Introduction" (<https://web.archive.org/web/20111114122103/http://www.eecs.qmul.ac.uk/~ohearn/Algol/intro.html>). Archived from the original (<http://www.eecs.qmul.ac.uk/~ohearn/Algol/intro.html>) on 14 November 2011.
4. "The ALGOL Programming Language" (<http://groups.engin.umd.umich.edu/CIS/course.des/cis400/algol/algol.html>) Archived (<https://web.archive.org/web/20161006113915/http://groups.engin.umd.umich.edu/CIS/course.des/cis400/algol/algol.html>) 6 October 2016 at the [Wayback Machine](#), University of Michigan-Dearborn
5. Backus, J. W.; Bauer, F. L.; Green, J.; Katz, C.; McCarthy, J.; Perlis, A. J.; Rutishauser, H.; Samelson, K.; Vauquois, B.; Wegstein, J. H.; van Wijngaarden, A.; Woodger, M. (May 1960). Naur, Peter (ed.). *Report on the Algorithmic Language ALGOL 60*. Copenhagen. doi:10.1145/367236.367262 (<https://doi.org/10.1145%2F367236.367262>). ISSN 0001-0782 (<https://www.worldcat.org/issn/0001-0782>).
6. "Revised Report on the Algorithmic Language Algol 60" (<http://www.masswerk.at/algol60/report.htm>). 1963. Archived (<https://web.archive.org/web/20070625171638/http://www.masswerk.at/algol60/report.htm>) from the original on 25 June 2007. Retrieved 8 June 2007.
7. "Revised Report on the Algorithmic Language ALGOL 68" (<https://www.cs.ru.nl/~hubbers/courses/sl1/rr.pdf>) (PDF). 1973. Archived (<https://web.archive.org/web/20140913132128/http://www.cs.ru.nl/~hubbers/courses/sl1/rr.pdf>) (PDF) from the original on 13 September 2014. Retrieved 13 September 2014.
8. Knuth, Donald E. (1964). "Backus Normal Form vs Backus Naur Form". *Communications of the ACM*. **7** (12): 735–736. doi:10.1145/355588.365140 (<https://doi.org/10.1145%2F355588.365140>).
9. ACM Award Citation: Peter Naur (<http://awards.acm.org/citation.cfm?id=1024454&srt=all&aw=140&ao=AMTURING&yr=2005>) Archived (https://wayback.archive-it.org/all/20120402220529/http://amturing.acm.org/award_winners/naur_1024454.cfm) 2 April 2012 at [Archive-It](#), 2005
10. "Hints on Programming Language Design" (http://www.eecs.umich.edu/~bchandra/courses/papers/Hoare_Hints.pdf) Archived (https://web.archive.org/web/20090915033339/http://www.eecs.umich.edu/~bchandra/courses/papers/Hoare_Hints.pdf) 15 September 2009 at the [Wayback Machine](#), C.A.R. Hoare, December 1973. Page 27. (This statement is sometimes erroneously attributed to [Edsger W. Dijkstra](#), also involved in implementing the first ALGOL 60 compiler.)
11. Dybvig, R. K.; et al. Rees, Jonathan; Clinger, William; Abelson, Hal (eds.). "Revised(3) Report on the Algorithmic Language Scheme, (Dedicated to the Memory of ALGOL 60)" (http://groups.csail.mit.edu/mac/ftplib/scheme-reports/r3rs-html/r3rs_toc.html). Archived (https://web.archive.org/web/20100114060759/http://groups.csail.mit.edu/mac/ftplib/scheme-reports/r3rs-html/r3rs_toc.html) from the original on 14 January 2010. Retrieved 20 October 2009.
12. Peter O'Hearn and Robert D. Tennent. 1996. *Algol-Like Languages*. Birkhauser Boston Inc., Cambridge, MA, USA.
13. "The Encyclopedia of Computer Languages" (<https://web.archive.org/web/20110927014141/http://hopl.murdoch.edu.au/showlanguage.prx?exp=1807>). Archived from the original (<http://hopl.murdoch.edu.au/showlanguage.prx?exp=1807>) on 27 September 2011. Retrieved 20 January 2012.
14. Computer Museum History (http://www.computerhistory.org/projects/zuse_z23/) Archived (https://web.archive.org/web/20100820213805/http://www.computerhistory.org/projects/zuse_z23/) 20 August 2010 at the [Wayback Machine](#), Historical Zuse-Computer Z23, restored by the Konrad Zuse Schule in Hünfeld, for the Computer Museum History Center in Mountain View (California) USA

15. Daylight, E. G. (2011). "Dijkstra's Rallying Cry for Generalization: the Advent of the Recursive Procedure, late 1950s – early 1960s" (<http://www.dijkstrascry.com/node/4>). *The Computer Journal*. **54**: 1756–1772. CiteSeerX 10.1.1.366.3916 (<https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.366.3916>). doi:10.1093/comjnl/bxr002 (<https://doi.org/10.1093%2Fcomjnl%2Fbxr002>). Archived (<https://web.archive.org/web/20130312111503/http://www.dijkstrascry.com/node/4>) from the original on 12 March 2013.
16. Kruseman Aretz, F.E.J. (30 June 2003). "The Dijkstra-Zonneveld ALGOL 60 compiler for the Electrologica X1". *Software Engineering* (<http://oai.cwi.nl/oai/asset/4155/04155D.pdf>) (PDF). History of Computer Science. Kruislaan 413, 1098 SJ Amsterdam: Centrum Wiskunde & Informatica. Archived (<https://web.archive.org/web/20160304191208/http://oai.cwi.nl/oai/asset/4155/04155D.pdf>) (PDF) from the original on 4 March 2016.
17. Hoare, Antony (1980). "The Emperor's Old Clothes" (http://amturing.acm.org/award_winners/hoare_4622167.cfm). *Communications of the ACM*. **24** (2). doi:10.1145/358549.358561 (<https://doi.org/10.1145%2F358549.358561>). Archived (http://archive.wikiwix.com/cache/20170913090744/http://amturing.acm.org/award_winners/hoare_4622167.cfm) from the original on 13 September 2017.
18. Koffman, Eliot. "All I Really Need to Know I Learned in CS1" (<https://web.archive.org/web/20121012032624/http://www.temple.edu/cis/directory/tenure/documents/KoffmanSIGCSESlides.pdf>) (PDF). Archived from the original (<http://www.temple.edu/cis/directory/tenure/documents/KoffmanSIGCSESlides.pdf>) (PDF) on 12 October 2012. Retrieved 20 May 2012.
19. "GOGOL - PDP-1 Algol 60 (Computer Language)" (<http://hopl.info/showlanguage.prx?exp=3905>). Online Historical Encyclopaedia of Programming Languages. Archived (<https://web.archive.org/web/20180202074636/http://hopl.info/showlanguage.prx?exp=3905>) from the original on 2 February 2018. Retrieved 1 February 2018.
20. Aho, Alfred V.; Sethi, Ravi; Ullman, Jeffrey D. (1986). *Compilers: Principles, Techniques, and Tools* (1st ed.). Addison-Wesley. ISBN 0-201-10194-7., Section 7.5, and references therein
21. "803 ALGOL" (<http://www.billp.org/ccs/A104/>) Archived (<https://web.archive.org/web/20100529063048/http://www.billp.org/ccs/A104/>) 29 May 2010 at the Wayback Machine, the manual for Elliott 803 ALGOL
22. "ICL 1900 series: Algol Language" (<http://www.icl1900.co.uk/techpub/tp3340.djvu>). ICL Technical Publication 3340. 1965.
23. How ASCII Got Its Backslash (<http://www.bobbemer.com/BACSLASH.HTM>) Archived (<https://web.archive.org/web/20140711225835/http://bobbemer.com/BACSLASH.HTM>) 11 July 2014 at the Wayback Machine, Bob Emer
24. iron/runic cross (<https://www.fileformat.info/info/unicode/char/16ed/>)
25. Decimal Exponent Symbol (<http://mailcom.com/unicode/DecimalExponent.ttf>)
26. Baumann, R. (October 1961). "ALGOL Manual of the ALCOR Group, Part 1" [ALGOL Manual of the ALCOR Group]. *Elektronische Rechenanlagen* (in German): 206–212.
27. Baumann, R. (December 1961). "ALGOL Manual of the ALCOR Group, Part 2" [ALGOL Manual of the ALCOR Group]. *Elektronische Rechenanlagen* (in German). **6**: 259–265.
28. Baumann, R. (April 1962). "ALGOL Manual of the ALCOR Group, Part 3" [ALGOL Manual of the ALCOR Group]. *Elektronische Rechenanlagen* (in German). **2**.
29. "GOST 10859 standard" (<https://web.archive.org/web/20070616201227/http://homepages.cwi.nl/~dik/english/codes/stand.html>). Archived from the original (<http://homepages.cwi.nl/~dik/english/codes/stand.html#gost10859>) on 16 June 2007. Retrieved 5 June 2007.

30. Broukhis, Leonid (22 January 2008). "Revised proposal to encode the decimal exponent symbol" (<https://www.unicode.org/L2/L2008/08030r-subscript10.pdf>) (PDF). *www.unicode.org*. ISO/IEC JTC 1/SC 2/WG 2. Archived (<https://web.archive.org/web/20150731024347/http://www.unicode.org/L2/L2008/08030r-subscript10.pdf>) (PDF) from the original on 31 July 2015. Retrieved 24 January 2016. "This means that the need to transcode GOST-based software and documentation can still arise: legacy numerical algorithms (some of which may be of interest,e.g. for the automatic landing of the Buran shuttle ...) optimized for the non-IEEE floating point representation of BESM-6 cannot be simply recompiled and be expected to work reliably, and some human intervention may be necessary."

Further reading

- F.L. Bauer, R. Baumann, M. Feliciano, K. Samelson, *Introduction to Algol*. Prentice Hall, 1964, ISBN 0-13-477828-6
- Brian Randell and L. J. Russell, *ALGOL 60 Implementation: The Translation and Use of ALGOL 60 Programs on a Computer*. Academic Press, 1964. The design of the **Whetstone Compiler**. One of the early published descriptions of implementing a compiler. See the related papers: *Whetstone Algol Revisited* (<http://www.cs.ncl.ac.uk/research/pubs/articles/papers/427.pdf>), and *The Whetstone KDF9 Algol Translator* (<https://web.archive.org/web/20100525044658/http://www.cs.ncl.ac.uk/publications/books/papers/124.pdf>) by Brian Randell
- Dijkstra, E. W (1961), *Algol 60 translation: an algol 60 translator for the x1 and making a translator for algol 60* (<http://www.cs.utexas.edu/users/EWD/MCReps/MR35.PDF>) (PDF), report MR 35/61, Amsterdam: Mathematisch Centrum
- Revised Report on the Algorithmic Language Algol 60 (<http://www.masswerk.at/algol60/report.htm>) by Peter Naur, et al. ALGOL definition
- "The European Side of the Last Phase of the Development of ALGOL 60" by Peter Naur (http://portal.acm.org/ft_gateway.cfm?id=808370&type=pdf&coll=&dl=ACM&CFID=15151515&CFTOKEN=6184618)

External links

- History of ALGOL (<http://www.softwarepreservation.org/projects/ALGOL/>) at the *Computer History Museum*
 - Web enabled ALGOL-F compiler for small experiments (<https://www.vintagebigblue.org/Compiler/ALGOLF/mvsAlgolFCompile.php>)
-

Retrieved from "<https://en.wikipedia.org/w/index.php?title=ALGOL&oldid=968530664>"

This page was last edited on 19 July 2020, at 23:27 (UTC).

Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.